C Global Surveyor

1. Overview

The goal of the CGS project is to demonstrate that it is possible to develop software verification tools that can analyze NASA programs using static analysis techniques to find a certain class of errors called runtime errors. First, we used a state-of-the-art commercial tool (PolySpace C-Verifier) and showed that it can successfully find errors in real NASA software (such as the flight software systems for Mars Path-Finder, Deep Space One, and Mars Exploration Rover). Second, we used the results of this first experience to design a research prototype (CGS) that addresses the shortcomings (in terms of precision and scalability) of the current commercial products.

The CGS tool analyzes C programs to find runtime errors. In brief, CGS analyzes every instruction in the source code of a C program to "check" if the operations performed in the instruction can create a problem at runtime (i.e., when the program will be executed). Specifically, CGS "checks" each instruction in a program to determine if the following problems can occur:

- **access to non-initialized variables:** the program attempts to used/read a variable that has not yet been assigned a value;
- **de-references of null pointers:** the program attempts to access the memory location referenced by the pointer even though the reference points to no memory location;
- **out-of-bound array accesses:** the program attempts to access an element of an a array using an index that is outside (strictly smaller or bigger than) the index bounds of the array.

CGS does its verification using static analysis techniques (i.e., techniques that do not require executing the code) based on the theory of Abstract Interpretation. Therefore, the analysis is performed at compile time (as opposed to runtime), and it does not require writing input cases. CGS analysis is conservative in the sense that it performs all checks necessary to find all errors. In most cases, CGS can certify that a check is correct (no error on any path leading to the check), or incorrect (there is an error on some paths leading to the check), or irrelevant because the check corresponds to dead code (the code that cannot be executed). In some cases, CGS cannot certify the correctness of the check, in which case it issues a warning.

CGS can analyze any C program, but its analysis algorithms have been tuned to be very precise (i.e, less than 10 percent of the checks are warnings) on programs following the

"Mars Path-Finder" (MPF) legacy. No precision guarantee is given for other types of C programs. For the moment, CGS runs only on top of the Linux operating system. For performance reasons, the analysis performed by CGS can be distributed over several processors (all running Linux). The results are centralized within an SQL database.

2. Goals

The overall goal of the CGS project is to show NASA developers that software reliability can be significantly increased by using verification tool based on static program analysis. We therefore need to demonstrate that it is possible to develop precise, and scalable, software verification tools that can analyze NASA programs using static analysis techniques to find a certain class of errors called runtime errors.

In general, developers recognize the usefulness of static analysis, but do not use it in practice because either commercial static analyzers do not scale or they generate too many warnings (which still need to be checked using traditional methods such as code inspection or testing). Our goal is to design and build a research prototype that scales to real NASA software systems (in the hundreds of thousands of lines of code), and deliver precise results (less than 10 percent of the checks are warnings).

3. Objectives

- The first objective was to assess the strengths and weaknesses of current commercial static program analyzers when they are applied to some real, and representative, NASA software systems written in C.
- The second objective is to develop a research prototype that addresses these shortcomings and demonstrate that it indeed scales and is precise for the chosen C programs (i.e., the MPF legacy software systems).
- The third objective is to study how the prototype performs on other C programs developed at NASA (for the International Space Station for example) and extend the algorithms to cover these cases.
- The fourth objective is to study how we can carry these experience from C programs to C++ programs so that we can address the next-generation of NASA software.

4. NASA Application

C Global Surveyor can be used in different parts of NASA. Our primary customer is the Jet Propulsion Lab, and in particular, its Mars program. Thus, CGS has been applied to the flight software of the Mars Path-Finder mission (135K lines of code), and of the Deep Space One mission (280K lines of code); CGS will be applied shortly to the flight software of the Mars Exploration Rover mission (650K lines of code). We will also look at other JPL-based missions.

We also plan to apply CGS to C code developed at other NASA centers such as Johnson Space Center and Marshall Space Flight Center. For example, we will analyze the code developed for the Habitat Holding Rack for the International Space Station. We are actively seeking other examples of large C programs in these centers.

The CGS extension that will deal with C++ programs will be developed so that it can be used during the development of the Mars Science Laboratory mission (MSL). Currently, MSL plans on using the Mission Data System (MDS) software platform which is implemented in C++. This tool will require significant advances in the analysis of pointers in the context of dynamic data structures.

5. Milestones

- Demonstrate an order of magnitude in verification cost improvement when using static analysis techniques over traditional techniques such as testing or code inspection -> Achieved in 2002.
- Analyze flight software for MPF-related missions with commercial tool:
  - Mars Path-Finder -> Achieved in May 2002
  - Deep Space One -> Achieved in September 2002
  - Mars Exploration Rover -> Achieved in June 2003
- Design and build CGS prototype -> Achieved in May 2003
- Analyze flight software for MPF-related missions with CGS:
  - Mars Path-Finder -> Achieved in June 2003
  - Deep Space One -> Achieved in July 2003
  - Mars Exploration Rover -> Due date March 2004
- Analyze other NASA C code:
  - Habitat Holding Rack -> Due date June 2004
- Identify a verifiable (by static analysis techniques) subset of C++ -> Due date June 2004
- Design extension for object-oriented framework to support static analysis -> Due date December 2004
- Build prototype for static analysis of (C++) MDS programs -> Due date April 2005
- Demonstrate (and measure performance of) analysis of a to-be-determined MDS adaptation -> Due date September 2005

Publications

Last changed: 20 December 2004 by Raymond De Ocampo